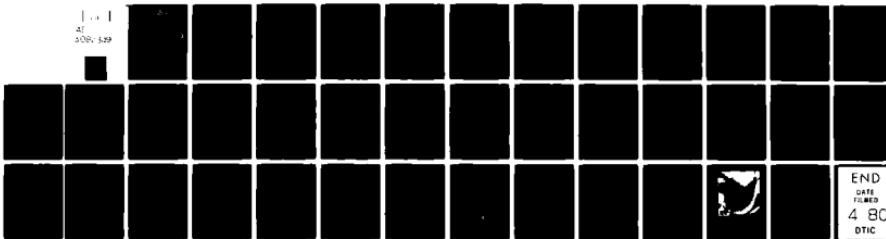


AD-A082 349 COLORADO STATE UNIV FORT COLLINS DEPT OF ELECTRICAL --ETC F/8 12/1
ASPECTS OF DYNAMIC PROGRAMMING IN SIGNAL AND IMAGE PROCESSING, (U)
NOV 79 L L SCHARF, H ELLIOTT DAAG29-79-C-0176

UNCLASSIFIED

NL

| . . |
AF
500V 529



END
FILED
4 80
DTIC

ADA082349

LEVEL

(12)
f

6 Aspects of Dynamic Programming in
Signal and Image Processing

10 Louis L. Scharf Senior Member IEEE
Howard Elliott Member IEEE

Manuscript Submitted to
IEEE Transactions on Automatic Control
BELLMAN SECTION

DTIC
SELECTED
MAR 24 1980

11 30 November 28, 1979

12 39

This document has been approved
for public release and sale; its
distribution is unlimited.

1 (15)
This work supported in part by the Army Research Office, Research Triangle Park, NC, under contract DAAG29-79-C-0176, and by the Office of Naval Research, Statistics and Probability Branch, Arlington, VA under contract N00014-75-C-0518. A preliminary version of this work was presented in an invited session on Signal Processing and Control Interactions at the Thirteenth Annual Asilomar Conference on Circuits, Systems and Computers, Nov. 5-7, 1979.

2
The authors are with the Electrical Engineering Department, Colorado State University, Fort Collins, CO 80523.

80 3 3 080

Abstract

The techniques peculiar to dynamic programming have found a variety of successful applications in the theory and practice of modern control. Successes in the theory and practice of signal and image processing are less numerous and prominent, but they do exist. In this paper we sound a call for renewed attention to the potential of dynamic programming for solving knotty nonlinear filtering problems in signal and image processing, and outline successes we have recently enjoyed in nonlinear frequency tracking and random boundary estimation in noisy black and white images. Two classical results, the fast Fourier transform (FFT) and Levinson's recursion for determining autoregressive parameters, are treated in the context of dynamic programming simply to reinforce our view that many of the algorithms we take for granted, and which were derived without recourse to dynamic programming, can be nicely interpreted as dynamic programming algorithms.

A

Accession For	
NTIS GRA&I	<input checked="" type="checkbox"/>
DDC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	<input type="checkbox"/>
By	
Distribution	
Approved by	
Distr	4.67 sec or Secular

I. Introduction

In a recent paper, Willsky [1] staked out some of the common ground upon which specialists in control and signal processing stand. A 1974 paper by Kailath [2] achieved a similar rapprochement between the classical frequency domain techniques associated with Wiener filtering and signal processing, and the time domain techniques associated with Kalman filtering and modern control. Such efforts have kindled interest in further exploring the territory common to the respective communities for the adventure and profit that is in it.

In this paper it is our aim to further stimulate this interest by showing that dynamic programming, a fundamental technique in control theory since Bellman's introduction and advocacy of it in the mid-1950's, can be of considerably more value in signal and image processing than has generally been recognized. This is not to say others have not recognized the potential of dynamic programming and exploited its techniques to solve interesting signal processing problems. We mention in particular Viterbi's dynamic programming algorithm for decoding convolutional code sequences [3], Cahn's dynamic programming algorithm for FM demodulation [4], and Forney's discussion of the Viterbi algorithm and other inference problems on finite-state Markov sequences that can be solved with the techniques of dynamic programming [5].

In the sections to follow we re-derive classical algorithms in discrete Fourier analysis and linear prediction using the principle of dynamic programming. We then present two new dynamic programming algorithms. One is for nonlinear frequency tracking and the other is for edge detection in noisy black and white images.

The organization is as follows. In Section II we use dynamic

programming arguments to re-derive the Goertzel and decimation-in-frequency FFT algorithms for efficiently computing the DFT. In Section IV we discuss the connections between control, detection, estimation, and prediction of autoregressive sequences observed in additive noise. We highlight the central role played by the so-called normal equations and re-derive the Levinson algorithm for recursively solving them in $O(N^2)$ operations. Sections V and VI contain the new results. In Section V we derive a dynamic programming algorithm for tracking a frequency sequence in additive noise. This is a rather classical nonlinear filtering problem. The results of Section VI show how dynamic programming may be used to derive a new algorithm for estimating local segments of object boundaries in noisy black and white images.

II. A Dynamic Programming Formalism

Traditionally, dynamic programming has been used to find "optimum" solutions to multi-stage decision problems. An "optimum" solution has generally been one that maximizes or minimizes a performance or cost functional. When the multi-stage decision problem is cast in a probabilistic framework the cost functional is typically a multivariable likelihood function or some monotone function of it.

Here is a formalism that is rich enough to embrace most of the "signal-in-noise" problems encountered in signal and image processing.

Let $\{x_k\}_{-\infty}^{\infty}$ denote a process with state variable representation

$$x_{k+1} = f_k(x_k, u_k)$$

$$y_k = g_k(x_k)$$

Here f_k and g_k may be random functions; the sequence $\{u_k\}$ is a parameter, decision, or control sequence that may be functionally dependent on the measurement sequence $\{y_k\}$. The range spaces for the state x_k , the parameter u_k , and the measurement y_k are respectively X , U , Y . These spaces may be finite, countable, or noncountable. When the spaces X and U are countable then their respective elements may be placed in one-to-one correspondence with the integers and the formalism of Markov chain theory may be mined. Even though the states of X may appear pretty uninteresting (the integers $0, 1, 2, \dots$), the mapping g_k may be chosen so that the signal component of $g_k(\cdot)$ generates characters c_k that are of great interest. The idea is simply to let a Markov chain control the dynamical state of the problem and reserve the role of character generation for the observation mechanism $g_k(\cdot)$. This point is illustrated in Figure 1.

As an example of character generation, consider the frequency tracking problem to follow in Section V. The state x_k evolves according to the model

$$x_{kN} = x_{(k-1)N} + v_{kN} ; \quad v_{kN} \in \{0, 1, \dots, Q-1\}$$

$$x_k = x_{k(\text{mod } N)} ; \quad x_k \in \{0, 1, \dots, Q-1\}$$

where $\{v_{kN}\}$ is a sequence of iid random variables.

Thus as k runs from 0 to K , x_k rests in state m for N steps, jumps to state n where it rests for N steps and so forth. This is illustrated in Figure 2. The character generation is defined by

$$g_k(x_k) = e^{jkx_k}$$

So, while the state rests in state m for N steps, the sequence of characters is a rotating sequence of phasors as illustrated in Figure 2.

Now consider a finite version of the process $\{x_k\}_{k=0}^{\infty}$:

$$X^N = (x_0, x_1, \dots, x_N)$$

$$= F_N(X^{N-1}, U^{N-1})$$

$$U^N = (u_0, \dots, u_N)$$

$$Y^N = (y_0, y_1, \dots, y_N)$$

Typically one wants to maximize a performance criterion

$$L_N(X^N, U^N, Y^N)$$

with respect to U^N , subject to constraints $C_N(X^N, U^N) = 0$. Call

$\hat{L}_N(X^N, \hat{U}^N, Y^N)$ the maximum. When \hat{L}_N obeys a recursion of the form

$$\hat{L}_N(X^N, \hat{U}^N, Y^N) = \hat{L}_{N-1}(X^{N-1}, \hat{U}^{N-1}, Y^N) + P_N(x_N, u_N, y_N) ,$$

then dynamic programming comes to the fore and the solution \hat{U}^N may be generated recursively as the limit of the following sequence of solutions:

$$\hat{U}^n = S_n(\hat{U}^{n-1}, Y^n) , \quad n=1, 2, \dots, N$$

Thus the central theme is to imbed the solution to an N stage problem in a sequence of simpler K stage problems. When the underlying state and parameter spaces are finite, the solution algorithm is finite-dimensional and implementable on a digital computer. When they are

uncountable, but the function L is quadratic, then it is still often possible to find a closed form recursive solution that may be programmed.

A very large class of problems may be formulated as above. Two particularly noteworthy examples are the linear discrete-time quadratic regulator problem in deterministic and stochastic control, and Markov chain sequence estimation in additive noise. On the other hand there are a great number of problems that admit dynamic programming solutions, but which are not naturally formulated in the style above. The FFT and Levinson algorithms discussed in Sections III and IV are well-known recursive problem solutions that have been derived without relying on a maximization argument.

One of the points we wish to make is the following: recognizing that a solution is a limit of a sequence of approximants that may be recursively computed is perhaps more fundamental than the search for a corresponding optimization problem. The chief value of an optimization formulation is that it often simplifies the search for the recursive solution algorithm.

III. Dynamic Programming, the DFT, and the FFT

The DFT certainly constitutes one of the cornerstones of modern Fourier analysis. Its uses range over the entire spectrum (so to speak) of signal processing applications. The DFT is a mapping, $DFT: \{x_n\}_0^{N-1} \rightarrow \{X_m\}_0^{N-1}$, that takes the sequence $\{x_n\}_0^{N-1}$ into the sequence $\{X_m\}_0^{N-1}$ according to the rule

$$X_m = \sum_{k=0}^{N-1} x_n w_N^{mn}, \quad m=0,1,\dots,N-1$$

$$w_N = e^{-j2\pi/N}$$

Noting that $w_N^{-mN} = 1 \forall m$, we may write X_m as follows:

$$X_m = \sum_{n=0}^{N-1} x_n w_N^{-m(N-n)}$$

This calculation may be viewed as the limit of the following sequence of imbedded approximations:

$$X_m^{(k)} = \sum_{n=0}^{k-1} x_n w_N^{-m(k-n)}, \quad k=1,2,\dots,N$$

Note $X_m^{(k)}$ obeys the following recursion:

$$X_m^{(k+1)} = w_N^{-m} X_m^{(k)} + w_N^{-m} x_k$$

$$X_m^{(N)} = X_m; \quad X_m^{(1)} = x_0 w_N^{-m}$$

This is the so-called Goertzel algorithm for obtaining the m th DFT variable, X_m , as the output of a digital filter excited by the sequence $\{x_n\}_0^{N-1}$. The output of the filter is read at time $k=N$. See Figure 3.

Dynamic Programming and the Decimation-in-frequency FFT

The Goertzel algorithm is a nice dynamic programming-like solution for the DFT. However it is not efficient. Let's see if we can improve upon it. Consider $X_m^{(k)}$ for even frequency indices $m=2r$:

$$x_{2r}^{(k)} = \sum_{n=0}^{k-1} x_n W_N^{-2r(k-n)}, \quad k=1, 2, \dots, N$$

$$= \sum_{n=0}^{k-1} x_n W_{N/2}^{-r(k-n)}, \quad k=1, 2, \dots, N$$

For k even (say $k=2s$),

$$\begin{aligned} x_{2r}^{(2s)} &= \sum_{n=0}^{2s-1} x_n W_{N/2}^{-r(2s-n)} \\ &= \sum_{n=0}^{s-1} x_n W_{N/2}^{-r(2s-n)} + \sum_{n=s}^{2s-1} x_n W_{N/2}^{-r(2s-n)} \\ &= W_{N/2}^{-rs} \sum_{n=0}^{s-1} x_n W_{N/2}^{-r(s-n)} + \sum_{\ell=0}^{s-1} x_{\ell+s} W_{N/2}^{-r(s-\ell)} \\ &= W_{N/2}^{-rs} x_{2r}^{(s)} + y_{2r}^{(s)} \end{aligned}$$

This shows that the $2s$ -point DFT approximant $x_{2r}^{(2s)}$ may be obtained from two s -point approximants. By choosing $s=N/2$ and continuing backwards in this way (for odd sub-indices, as well) one arrives at a backward dynamic programming derivation of the decimation-in-frequency FFT. See Figure 4 for an elementary representation of a 4-point decimation in frequency FFT.

A more classical dynamic programming derivation would proceed from the minimization of the quadratic form

$$Q_N = \sum_{n=0}^{N-1} |x_n - \sum_{m=0}^{N-1} x_m e^{j2\pi \frac{mn}{N}}|^2$$

with respect to $\{x_m\}_0^{N-1}$.

IV. Detection, Estimation, and Control Structures

in the AR(N) Case: Kalman Filters, Levinson Recursions, and Dynamic Programming

Autoregressive (AR) models for signals, states, and data play a starring role in many areas of signal processing and control. By appropriately selecting model parameters (and order) one can model the covariance structure and spectral characteristics of more general models. The so-called normal equations for identifying AR parameters are elegant and easily solved with recursions of the Levinson-type.

In this section we tie up control, prediction, detection, and estimation in the special case where we are dealing with a zero-mean, wide-sense stationary, scalar autoregressive time series. The usual state-variable and matrix block diagrams give way to scalar variables and digital filter blocks of moving average filters. The normal equations are highlighted and dynamic programming is used to derive the famous Levinson recursions.

Models

Let $\{x_k\}$ denote a scalar zero-mean, wide-sense stationary autoregressive sequence that obeys the recursion

$$x_k = \sum_{n=1}^N a_n x_{k-n} + w_k \quad k$$

w_n : sequence of i.i.d. $N(0, \sigma_w^2)$ r.v.s.

It is easy to see that the covariance sequence $\{r_m\}_{m=-\infty}^{\infty}$, $r_m = r_{-m}$, associated with the sequence $\{x_k\}$ obeys the recursion

$$r_m = \sum_{n=1}^N a_n r_{m-n} + \sigma_w^2 \delta_m, \quad m=0,1,\dots,$$

From here one may write out the so-called normal equations:

$$\begin{bmatrix} r_0 & r_1 & \cdots & r_{N-1} \\ r_1 & r_0 & r_1 & \cdots & r_{N-2} \\ \vdots & & & & \\ \vdots & & & r_1 & \\ r_{N-1} & r_1 & r_0 & & \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ \vdots \\ a_N \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ \vdots \\ r_N \end{bmatrix}$$

If the $\{a_n\}_1^N$ are known these equations are used to solve for the $\{r_n\}_{0}^N$ [6]. Conversely, if the $\{r_n\}_0^N$ are known, these equations are used to solve for the $\{a_n\}_1^N$.

In much of what follows we will assume the sequence $\{x_k\}$ is observed in zero-mean additive WGN:

$$z_k = x_k + n_k \quad \forall k$$

n_k : sequence of i.i.d. $N(0, \sigma_n^2)$ r.v.s.

The companion form state model for all of this follows:

$$x_{k+1} = A x_k + b u_k$$

$$z_k = C' x_k$$

$$x_k = \begin{bmatrix} x_{k-N+1} \\ \vdots \\ x_{k-1} \\ x_k \end{bmatrix}, \quad A = \begin{bmatrix} 0 & & & \\ & I & & \\ 0 & & & \\ \hline a_N & a_{N-1} & \cdots & a_0 \end{bmatrix}, \quad b=C=\begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}, \quad u_k=w_{k+1}$$

Noisy Prediction and the Kalman Predictor

The stationary Kalman one-step predictor for the noisily observed AR(N) sequence is

$$\hat{x}_{k+1} = A \hat{x}_k + K(z_k - \hat{x}_k)$$

$$\hat{x}_k = C' \hat{x}_k$$

where

$$K = APC(C'PC + \sigma_n^2)^{-1}$$

$$= [k_1, k_2, \dots, k_N] \quad (1)$$

$$P = (A - KC') P (A - KC') + \sigma_n^2 KK' + \sigma_w^2 bb' \quad (2)$$

The Kalman prediction sequence $\{\hat{x}_{k+1}\}$ for $\{z_{k+1}\}$ can be interpreted as the output of an ARMA(N,N-1) filter, driven by the prediction error sequence $\{v_k = z_k - \hat{x}_k\}$ or an ARMA(N,N-1) filter driven by the observation sequence $\{z_k\}$. The resulting filter equations are

$$\hat{x}_{k+1} = \sum_{i=1}^N a_i \hat{x}_{k+1-i} + \sum_{i=1}^N g_i v_{k+1-i}$$

or

$$\hat{x}_{k+1} = \sum_{i=1}^N (a_i - g_i) \hat{x}_{k+1-i} + \sum_{i=1}^N g_i z_{k+1-i} \quad (3)$$

where the coefficients, g_i , can be defined by the characteristic polynomial $\Delta(\lambda)$ of $(A - KC')$:

$$\Delta(\lambda) = \lambda^N + \sum_{i=1}^N (g_i - a_i) \lambda^{N-i}$$

See Figure 5 for a block diagram of this predictor. Note that the noise-free MA predictor filter, $P(z) = \sum_{n=1}^N a_n z^{-n}$, is preserved in the feedback loop, but that the residual sequence $v_k = z_k - \hat{x}_k$ is now weighted with a feedforward MA filter, $Q(z) = \sum_{n=1}^N g_n z^{-n}$.

Why is the noisy Kalman predictor ARMA and not MA? The answer is that $\{z_k\}$, a noisy version of an AR signal process, obeys an ARMA(N,N) difference equation. As an AR(N) model has an MA(N-1) predictor, it is at least logical (if not intuitive) that an ARMA(N,N) process has an ARMA(N,N-1) predictor.

The Noise Free Predictor ($\sigma_n^2 = 0$)

The prediction vector \hat{x}_k consists of the terms

$$E[x_{k-N+1}/z_{k-1}, z_{k-2}, \dots]$$

.

$$E[x_{k-1}/z_{k-1}, z_{k-2}, \dots]$$

$$E[x_k/z_{k-1}, z_{k-2}, \dots]$$

When $\sigma_n^2 = 0$, then $z_k = x_k \forall k$ and

$$E[x_{k-n}/z_{k-n}, z_{k-n-1}, \dots]$$

$$= E[x_{k-n}/x_{k-n}, \dots] = x_{k-n}, n=1, 2, \dots$$

So in this case the prediction vector is

$$\hat{x}_k = \begin{bmatrix} x_{k-N+1} \\ x_{k-N+2} \\ \vdots \\ \vdots \\ x_{k-1} \\ \hat{x}_{k/k-1} \end{bmatrix}$$

It follows that P , the covariance $E[\hat{x}_k - x_k][\hat{x}_k - x_k]'$ is

$$P = \begin{bmatrix} 0 & \dots & 0 \\ \vdots & & \vdots \\ \vdots & & 0 \\ 0 & \dots & 0 \end{bmatrix} \quad (4)$$

Calculating K by substituting (4) into (1), we find that $\Delta(\lambda) = \lambda^N$

and hence $g_1 = a_1$. This implies, as one would expect, that the prediction filter (3) reduces to the purely moving average relation

$$\hat{x}_{k+1} = \sum_{i=1}^N a_i z_{k+1-i} .$$

Minimum Variance Control

One of the simplest control strategies is minimum variance regulation where one desires to minimize the variance of the AR(N) output sequence $\{x_k\}$, and force $E(x_k) = 0$. The well known separation principle

allows one to generate a feedback control strategy assuming noisefree measurements, i.e. $n_k = 0$, and then use the same strategy in the noisy case but with the Kalman filter estimates $\{\hat{x}_k\}$ replacing the actual filter outputs $\{x_k\}$.

Assume then we have the system

$$x_k = \sum_{i=1}^N a_i x_{k-i} + w_k + v_k$$

where $\{v_k\}$ is our feedback control sequence. We would like to minimize

$$\begin{aligned} E(x_k^2) &= E\left(\sum_{i=1}^N a_i x_{k-i} + w_k + v_k\right)^2 \\ &= E(w_k^2 + 2w_k v_k + 2w_k \left(\sum_{i=1}^N a_i x_{k-i}\right) + (v_k + \sum_{i=1}^N a_i x_{k-i})^2) \\ &= E(w_k^2) + 2E(w_k v_k | v_k) E(v_k) \\ &\quad + 2E(w_k \left(\sum_{i=1}^N a_i x_{k-i}\right)) + E((v_k + \sum_{i=1}^N a_i x_{k-i})^2). \end{aligned}$$

Since $\{w_k\}$ is uncorrelated with $\{x_{k-i}\}$, $i \geq 1$, and since $E(w_k v_k | v_k) = 0$, it is clear that $E(x_k^2)$ is minimized by choosing

$$v_k = -\sum a_i x_{k-i}$$

This control is illustrated in Figure 5 as a feedback loop running up the left side of the figure. The feedback loop to the top COMPUTE box shows how \hat{x}_k would be used for minimum variance control in the noisy case.

Detection and the Likelihood Ratio

Consider the hypothesis test H_0 vs. H_1 with

$$H_0 : z_k = n_k , \quad k=0,1,\dots,K$$

$$H_1 : z_k = x_k + n_k , \quad k=0,1,\dots,k$$

This test is equivalent to the test \hat{H}_0 vs. \hat{H}_1 where

$$\hat{H}_0 : v_k^{(0)} = z_k : N(0, \sigma_n^2)$$

$$\hat{H}_1 : v_k^{(1)} : N(0, p_0 + \sigma_n^2) ; p_0 : \text{variance of } \hat{x}_k$$

and $v_k^{(1)} = z_k - \hat{x}_k$ is the innovations sequence in the Kalman filter. The log-likelihood ratio for this problem is [7]

$$LR = K - \frac{1}{p_0 + \sigma_n^2} \sum_{k=0}^K v_k^{(1)} + \frac{1}{\sigma_n^2} \sum_{k=0}^K z_k^2$$

Thus the statistics $\sum v_k^{(1)}$ and $\sum z_k^2$ are sufficient and the log-likelihood ratio may be computed as in Figure 5.

The Normal Equations are Fundamental

It should be clear from Figure 5 that the AR coefficients $\{a_n\}_1^N$ characterizing the sequence $\{x_k\}$ are fundamental to the implementation of control, prediction, and detection algorithms on noisily observed AR sequences. Unfortunately, sequences rarely come tagged with their corresponding AR parameters. More typically finite records of them come to us and we estimate a covariance function (or power spectrum), often by FFT-ing, squaring and windowing, and inverse FFT-ing. These estimates may then be used to solve for the coefficients $\{a_n\}_1^N$ from the normal equations

$$\begin{bmatrix} r_0 & \dots & r_{N-1} \\ r_1 & r_0 & \dots \\ \vdots & \vdots & \vdots \\ r_{N-1} & \dots & r_0 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_N \end{bmatrix} \quad (4)$$

This makes the normal equations fundamental and arouses our interest in efficient ways of solving them. The derivation that follows was motivated by Bellman's discussion in [8].

Dynamic Programming and Levinson's Algorithm

Rewrite the normal equations as

$$\sum_{m=1}^N \alpha_m^N r_{|n-m|} = r_n , \quad n=1, 2, \dots, N$$

These equations characterize the $\{\alpha_m\}_1^N$ that minimize the quadratic form

$$Q_N = r_0 - 2 \sum_{m=1}^N \alpha_m r_m + \sum_{m=1}^N \sum_{n=1}^N \alpha_m \alpha_n r_{|n-m|}$$

The minimum is

$$Q_N^N(r_1, \dots, r_N) = r_0 - \sum_{m=1}^N \alpha_m^N r_m \quad (4b)$$

Assuming the normal equations to be non-singular, we may write

$$\alpha_m^N = \sum_{n=1}^N S_{mn}^N r_n , \quad m=1, 2, \dots, N \quad (5)$$

$$Q_N^N(r_1, \dots, r_N) = r_0 - \sum_{m=1}^N \sum_{n=1}^N r_m S_{mn}^N r_n \quad (6)$$

Write out $Q_N(\cdot)$ as follows:

$$\begin{aligned} Q_N(r_1, \dots, r_N) &= r_0 - 2 \sum_{m=1}^{N-1} \alpha_m r_m - 2\alpha_N r_N + \sum_{m=1}^{N-1} \sum_{n=1}^{N-1} \alpha_m \alpha_n r_{|m-n|} \\ &\quad + 2 \sum_{m=1}^{N-1} \alpha_m \alpha_N r_{|N-m|} + \alpha_N^2 r_0 \\ &= \alpha_N^2 r_0 - 2\alpha_N r_N + Q_{N-1}(r_1 - \alpha_N r_{N-1}, \dots, r_{N-1} - \alpha_N r_1) \end{aligned}$$

So the minimization of $Q_N(r_1, \dots, r_N)$ with respect to $\{\alpha_m\}_1^N$ leads to

$$\begin{aligned} Q_N^N(r_1, \dots, r_N) &= \min_{\{\alpha_m\}_1^N} Q_N(r_1, \dots, r_N) \\ &= \min_{\alpha_N} [\alpha_N^2 r_0 - 2\alpha_N r_N + \min_{\{\alpha_m\}_1^{N-1}} Q_{N-1}(r_1 - \alpha_N r_{N-1}, \dots)] \\ &= \min_{\alpha_N} [\alpha_N^2 r_0 - 2\alpha_N r_N + Q_{N-1}^{N-1}(r_1 - \alpha_N r_{N-1}, \dots)] \quad (7) \end{aligned}$$

This equation contains the essence of dynamic programming and the

principle of optimality: Once the $\{\alpha_1, \dots, \alpha_{N-1}\}$ are found, Q_{N-1}^{N-1} may be constructed and α_N found as a function of r_0, r_N and $\{\alpha_1, \dots, \alpha_{N-1}\}$. One continues in this way. At each step of the way the minimization problem on $Q_{N-1}(\cdot)$ is quadratic.

Let's use (6) in the RHA of (7):

$$\begin{aligned} Q_N^N(r_1, \dots, r_N) &= \min_{\alpha_N} [\alpha_N^2 r_0 - 2\alpha_N r_N + r_0 \\ &\quad - \sum_{m=1}^{N-1} \sum_{n=1}^{N-1} (r_m - \alpha_N r_{N-m}) S_{mn}^{N-1} (r_n - \alpha_N r_{N-n})] \\ &= \min_{\alpha_N} [\alpha_N^2 (r_0 - \sum_{m=1}^{N-1} \sum_{n=1}^{N-1} r_{N-m} S_{mn}^{N-1} r_{N-n}) \\ &\quad - 2\alpha_N (r_N - \sum_{m=1}^{N-1} r_m \sum_{n=1}^{N-1} S_{mn}^{N-1} r_{N-n}) \\ &\quad + Q_{N-1}^{N-1}(r_1, \dots, r_{N-1})] \end{aligned}$$

It follows easily that the minimizing value of α_N is

$$\alpha_N = \frac{r_N - \sum_{m=1}^{N-1} \sum_{n=1}^{N-1} r_m S_{mn}^{N-1} r_{N-n}}{r_0 - \sum_{m=1}^{N-1} \sum_{n=1}^{N-1} r_{N-m} S_{mn}^{N-1} r_{N-n}}$$

Use (5) in the numerator to get

$$\alpha_N = \frac{r_N - \sum_{n=1}^{N-1} \alpha_n^{N-1} r_{N-n}}{r_0 - \sum_{m=1}^{N-1} \sum_{n=1}^{N-1} r_{N-m} S_{mn}^{N-1} r_{N-n}}$$

Let's call

$$c_n^{N-1} = \sum_{m=1}^{N-1} r_{N-m} S_{mn}^{N-1}, \quad n=1, 2, \dots, N-1$$

and see if we can find a recursion for it. In the meantime

$$\alpha_N^N = \frac{r_N - \sum_{n=1}^{N-1} \alpha_n^{N-1} r_{N-n}}{r_0 - \sum_{n=1}^{N-1} c_n^{N-1} r_{N-n}} \quad (8)$$

Note

$$\begin{aligned} \alpha_N^N (r_0 - \sum_{n=1}^{N-1} c_n^{N-1} r_{N-n}) & - 2\alpha_N^N (r_N - \sum_{n=1}^{N-1} \alpha_n^{N-1} r_{N-n}) \\ & = \alpha_N^N (r_N - \sum_{n=1}^{N-1} \alpha_n^{N-1} r_{N-n}) - 2\alpha_N^N (r_N - \sum_{n=1}^{N-1} \alpha_n^{N-1} r_{N-n}) \\ & = -\alpha_N^N (r_N - \sum_{n=1}^{N-1} \alpha_n^{N-1} r_{N-n}) \end{aligned}$$

So we have this recursion for the minimum prediction error:

$$Q_N^N(r_1, \dots, r_N) = Q_{N-1}^{N-1}(r_1, \dots, r_{N-1}) - \alpha_N^N (r_N - \sum_{n=1}^{N-1} \alpha_n^{N-1} r_{N-n})$$

Now use (4b) to get

$$r_0 - \sum_{m=1}^N \alpha_m^N r_m = r_0 - \sum_{m=1}^{N-1} \alpha_m^{N-1} r_m - \alpha_N^N r_N + \sum_{n=1}^{N-1} \alpha_N^N \alpha_{N-n}^{N-1} r_{N-n}$$

Or

$$\sum_{m=1}^{N-1} \alpha_m^N r_m = \sum_{m=1}^{N-1} (\alpha_m^{N-1} - \alpha_N^N \alpha_{N-m}^{N-1}) r_m$$

Thus from the recursion on Q_N^N we get this recursion for the α_m^N :

$$\alpha_m^N = \alpha_m^{N-1} - \alpha_N^N \alpha_{N-m}^{N-1}, \quad m=1, 2, \dots, M-1$$

Our one remaining problem is c_n^N . Write it as

$$c_m^N = \sum_{n=1}^N s_{mn}^N r_{N+1-n}$$

Thus c_m^N must be the solution of

$$\sum_{m=1}^N C_m^N r_{|n-m|} = r_{N+1-n}, \quad n=1, 2, \dots, N-1 \quad (9)$$

It therefore satisfies a recursion just like α_m^N . In fact (9) is just

(4a) with the RHS vector turned upside down. By the symmetry of

$R = \{r_{|m-n|}\}$, this simply turns the solution vector $\{\alpha_m^N\}$ upside down.

Thus

$$C_m^N = \alpha_{N+1-m}^N, \quad m=1, 2, \dots, N$$

and we are done.

Summary: The Levinson recursions may now be written

$$\alpha_N^N = \frac{r_N - \sum_{n=1}^{N-1} \alpha_n^{N-1} r_{N-n}}{r_0 - \sum_{n=1}^{N-1} \alpha_{N-n}^{N-1} r_{N-n}}$$

$$\alpha_m^N = \alpha_m^{N-1} - \alpha_N^{N-1} \alpha_{N-m}^{N-1}, \quad m=1, 2, \dots, N-1$$

$$Q_N^N = Q_{N-1}^{N-1} - \alpha_N^N (r_N - \sum_{m=1}^{N-1} \alpha_m^{N-1} r_{N-m})$$

Call $\alpha_n^N = a_n$, $n=1, 2, \dots, N$ to have the solution to the normal equations given in (4). To get these equations into their fully modern form, one must look at a backwards prediction.

V. Frequency Tracking and Dynamic Programming

Phase and frequency tracking problems comprise some of the most nettlesome nonlinear filtering problems in the entire realm of signal processing and communication. A typical problem is the following: observe the sequence $\{z_k\}$ with

$$z_k = s_k + n_k$$

$$s_k = e^{j\omega_k k}$$

and estimate the FM sequence $\{\omega_k\}$. Here $s_k = e^{j\omega_k k}$ is a randomly frequency modulated signal. The sequence $\{n_k\}$ is additive noise and $\{\omega_k\}$ is a sequence of angular frequencies. Assume $n_k : N(0, \sigma^2)$ (complex normal).

We wish to observe a record of data $\{z_k\}_{k=0}^{KN}$ and infer the most likely sequence of frequencies, $\{\hat{\omega}_k\}_{k=0}^{KN}$. For our model of $\{\omega_k\}$ we take each $\omega_k \in \{0, 2\pi/Q, \dots, 2\pi(Q-1)/Q\}$, evolving according to

$$\omega_{kN} = \omega_{(k-1)N} + v_{kN}$$

$$\omega_{kN+l} = \omega_{kN}, \quad l=0, 1, \dots, N-1 \quad \forall k$$

A typical sequence $\{\omega_k\}$ is illustrated in Figure 5. The independent increments sequence $\{v_{kN}\}$ is a sequence of i.i.d. r.v.s. selected in such a way that the transition probabilities

$$p(\omega_{kN}/\omega_{(k-1)N})$$

correspond to our notion of physical reality. We may think of the sequence $\{\omega_{kN}\}$ as a finite-state random walk on the circle with an unusual transition probability structure. Typical trajectories for $\{\omega_k\}$ and $\{s_k = e^{j\omega_k k}\}$ are illustrated in Figure 5.

Let's organize the sequence $\{z_k\}_{k=0}^{KN}$ into contiguous blocks of length N , of the form $\{z_{kN+l}\}_{l=0}^{N-1}$. See Figure 6. Recall

$$z_{kN+\ell} = e^{j\omega_{kN+\ell}(kN+\ell)} + n_{kN+\ell}$$

$$= e^{j\omega_{kN}(kN+\ell)} + n_{kN+\ell}$$

Write $\{z_k\}_0^{KN} = \sum_{k=0}^{K-1} \{z_{kN+\ell}\}_{\ell=0}^{N-1}$. Consider the joint density of $\{z_k\}_0^{KN}$ and $\{\omega_{kN}\}_0^{K-1}$:

$$f(\sum_{k=0}^{K-1} \{z_{kN+\ell}\}_{\ell=0}^{N-1}, \{\omega_{kN}\}_0^{K-1})$$

Exploit the probabilistic structure of $\{z_k\}$ and $\{\omega_{kN}\}$ to write this as

$$f(\cdot, \cdot) = \prod_{k=0}^{K-1} \prod_{\ell=0}^{N-1} N z_{kN+\ell} (e^{j\omega_{kN}(kN+\ell)}, \sigma_n^2) \prod_{k=0}^{K-1} p(\omega_{kN}/\omega_{(k-1)N})$$

The natural logarithm of $f(\cdot, \cdot)$ is proportional to

$$\ln f(\cdot, \cdot) \sim 2 \operatorname{Re} \frac{1}{2\sigma_n^2} \sum_{k=0}^{K-1} \sum_{\ell=0}^{N-1} z_{kN+\ell} e^{-j\omega_{kN}(kN+\ell)}$$

$$+ \sum_{k=0}^{K-1} \ln p(\omega_{kN}/\omega_{(k-1)N})$$

Write

$$\begin{aligned} \ln f(\cdot, \cdot) &\sim 2 \operatorname{Re} \frac{1}{2\sigma_n^2} \sum_{k=0}^{K-1} e^{-j\omega_{kN} kN} \sum_{\ell=0}^{N-1} z_{kN+\ell} e^{-j\omega_{kN} \ell} + \\ &+ \sum_{k=0}^{K-1} \ln p(\omega_{kN}/\omega_{(k-1)N}) \end{aligned}$$

Let $X_{kN}(\theta)$ denote the finite Fourier transform

$$X_{kN}(\theta) = \sum_{\ell=0}^{N-1} z_{kN+\ell} e^{j\theta \ell}$$

Then the log-likelihood function may be written

$$\begin{aligned} \ln f(\cdot, \cdot) &\sim \operatorname{Re} \frac{1}{2\sigma_n^2} \sum_{k=0}^{K-1} e^{-j\omega_{kN} kN} X_{kN}(\theta=\omega_{kN}) \\ &+ \sum_{k=0}^{K-1} \ln p(\omega_{kN}/\omega_{(k-1)N}) \end{aligned}$$

Our notion of the most likely sequence $\{\hat{\omega}_k\}_{0}^{KN}$ is the sequence

$$\hat{\omega}_{kN+\ell} = \hat{\omega}_{kN}, \quad \ell=0,1,2,\dots,N-1, \quad k=0,1,\dots,K-1$$

where $\{\hat{\omega}_k\}_{0}^{K-1}$ is the sequence that maximizes $f(\cdot, \cdot)$. Thus we consider the maximization problem

$$\max_{\{\omega_k\}_{0}^{K-1}} \frac{1}{\sigma_n^2} \operatorname{Re} \sum_{k=0}^{K-1} e^{-j\omega_k N^{kN}} X_{kN}(\theta=\omega_{kN}) + \sum_{k=0}^{K-1} \ell \ln p(\omega_{kN}/\omega_{(k-1)N})$$

Write this as

$$\max_{\{\omega_k\}_{0}^{K-1}} \Gamma_{K-1}$$

with Γ satisfying the following recursion:

$$\Gamma_t = \Gamma_{t-1} + \ell \ln p(\omega_{tN}/\omega_{(t-1)N}) + \frac{1}{\sigma_n^2} \operatorname{Re} e^{-j\omega_{tN} tN} X_{tN}(\theta=\omega_{kN})$$

So our maximization problem becomes

$$\begin{aligned} \max_{\{\omega_k\}_{0}^{K-1}} & [\max_{\{\omega_k\}_{0}^{K-3}} \Gamma_{K-2} + \ell \ln p(\omega_{(K-1)N}/\omega_{(K-2)N}) + \\ & + \frac{1}{\sigma_n^2} \operatorname{Re} e^{-j\omega_{(K-1)N} (K-1)N} X_{(K-1)N}(\theta=\omega_{(K-1)N})] \end{aligned}$$

This form leads to the following observation: the maximizing frequency trajectory, call it $\{\hat{\omega}_k\}$, passing through $\hat{\omega}_{(K-2)N}$ on its way to $\hat{\omega}_{(K-1)N}$, must arrive at $\hat{\omega}_{(K-2)N}$ along a route $\{\hat{\omega}_k\}_{0}^{K-3}$ that maximizes Γ_{K-2} . If it did not we could retain $\hat{\omega}_{(K-2)N}$ and $\hat{\omega}_{(K-1)N}$ and with a different sequence to get a larger Γ_{K-1} . It is this observation that forms the basis of forward dynamic programming.

Recall the $\omega_{kN} \in \{2\pi r/Q\}_{r=0}^{Q-1}$. This means the finite Fourier transform $X_{kN}(\theta)$ must only be evaluated on $\theta=2\pi r/Q$, $r=0,1,\dots,Q-1$. The best way to do this is to zero-pad $\{x_{kN+\ell}\}_{\ell=0}^{N-1}$ to obtain a Q-point sequence

that can be FFT'd to get $X_{kN}(\theta=2\pi r/Q)$. See Figure 5. Then for each node on Figure 5 we evaluate $X_{kN}(\theta=2\pi r/Q)$, and find the best route through the trellis with a dynamic programming algorithm. This completes our algorithm for moderating the usual peak-picking rule on the FFT with prior information $p(\omega_{kN}/\omega_{(k-1)N})$.

The reader is referred to [9] for a more complete discussion of a related algorithm for nonlinear phase tracking.

VI. Local Boundary Estimation in Noisy Black and White Images

In digital image processing one is interested in developing computer algorithms which can either automatically extract information from pictures or at least simplify the process of manually interpreting them. In either case, a basic step involves segmenting a picture into regions with similar features such as grey level or texture. This involves the estimation of region boundaries. Boundary estimation algorithms make use of operators which estimate short segments of boundaries using picture data in small picture sections. Examples are simple gradient operators or the well known Hueckel operator [10]. An example of a local sequential estimator which is also used for this purpose can be found in [11]. In this section we outline a new dynamic programming algorithm for sequentially estimating short boundary segments.

Image and Boundary Models

Let a digitized black and white image be represented by a matrix with components g_{ij} corresponding to the grey level value of a picture element (pixel) centered at position (i,j) . The value g_{ij} will have two components - a true picture component b_{ij} and a noise component n_{ij} so that $g_{ij} = b_{ij} + n_{ij}$. A picture is assumed to consist of a single region of grey level r_{in} lying in a background of grey level r_{out} , so that b_{ij} can take on either of the two values r_{in} or r_{out} . The noise components n_{ij} are assumed to be independent identically distributed Gaussian random variables with mean zero and variance σ^2 .

An edge element is defined as the line segment separating two adjacent pixels, and as shown in Figure 8 a boundary segment consists of a directed sequence of edge elements $\{t_{ij}\}_1^M$. As illustrated in

Figure 9 , we model short boundary edge sequences as being generated sequentially by passing out of rectangular boxes R_k , $k=1,2,\dots,N$, of size $\rho_k = kx_2(K-1)$. Figure 10a gives an example of a boundary which is consistent with this model while Figure 10b shows a similar but inconsistent boundary. In the latter case the edge sequence re-enters R_4 . Although this model restricts somewhat the types of boundary segments that can be generated, it is still very reasonable for region boundaries with low and slowly varying curvatures such as the Cat Scan Lung Section shown in Figure 11. The maximum rectangle size N is assumed fixed *a priori* and is a function of the boundary curvature properties for the region of interest.

Boundary segments generated by such a model are naturally represented by a sequence of states in a Markov chain where the index parameter k for the rectangle size ρ_k is also the index parameter for the Markov process. A process state, x_k , at "time" k , $k \geq 1$, will correspond geometrically to the end point of a boundary sequence passing out of R_k . Figure 12 shows the state locations, x_k^j , for $k=1,2,\dots,5$. Note that the number of possible states at time k is 1 for $k=1$, 3 for $k=2$, and $9 + 4(k-3)$ for $k \geq 3$.

Figure 13 contains an abstract representation of a typical realization of the Markov process, together with a description of the picture or character, c_k , associated with each state. The observed image will be a noise corrupted version of each such picture.

If the regions of interest have smooth, low curvature boundaries then a reasonable rule for assigning transition probabilities $p(x_k|x_{k-1})$ is to choose $p(x_k|x_{k-1})$ to be inversely related to the distance (measured in edge elements) between states x_{k-1} and x_k . We must also impose the

constraint that $\sum_{j=1}^{9+4(k-3)} p(x_k^j | x_{k-1}) = 1.$

Using this model, a boundary edge sequence $\{t_i\}_1^M$ will correspond to a state sequence $\{x_k\}_1^N$.

A Dynamic Programming Algorithm for Estimating Boundary Segments

Using the pixel data in an $N \times 2(N-1)$ block, R_N , we next formulate a dynamic programming algorithm for optimally estimating edge sequences that are generated by the previously described model. The algorithm is optimal in the sense that it maximizes the joint likelihood of an edge sequence through, and all the data in, the rectangle R_N .

To begin we first define the pixel data sets

$$D_k = \{g_{ij} : \text{pixel } (i,j) \in R_k\}$$

$$d_k = \{g_{ij} : \text{pixel } (i,j) \in R_k \text{ and pixel } (i,j) \notin R_{k-1}\}.$$

This implies that $D_k = D_{k-1} \cup d_k$, $D_1 = \text{empty set}$. This recursion is essential. Next let $\ln L(\cdot)$ denote a log-likelihood function, and $S_N = \{x_k\}_1^N$ denote a boundary state sequence of length N . Then $\ln L(D_N, S_N)$, the joint log-likelihood of a boundary state sequence and the picture data, must satisfy

$$\ln L(D_N, S_N) = \ln L(D_N | S_N) + \ln L(S_N) \quad (9)$$

where $\ln L(S_N)$ is the log-likelihood of the state sequence S_N and $\ln L(D_N | S_N)$ is the pixel data log-likelihood conditioned on the boundary $\{t_i\}_1^M$ described by S_N . Since the state sequence S_N is a Markov chain we can use

$$\ln L(S_N) = \ln L(S_{N-1}) + \ln p(x_N | x_{N-1}) \quad (10)$$

$$\ln L(S_1) = \ln L(x_1) = \ln p_s(x_1) \quad (11)$$

where $p_s(x_1)$ is the probability of a particular starting state x_1 .

Since boundary edge sequences are prohibited from re-entering rectangles from which they have already passed out of, we can express

$$\ln L(D_N | S_N) = \ln L(D_{N-1} | S_{N-1}) + \ln L(d_N | x_N) \quad (12)$$

where $\ln L(d_N | x_N)$ is the log-likelihood of the data added in extending the state sequence S_{N-1} to S_N conditioned on the specific new state x_N . Substitution of (12) and (10) into (9) leads to the following recursive expression for $L(D_N, S_N)$:

$$\ln L(D_N, S_N) = \ln L(D_{N-1}, S_{N-1}) + \ln p(x_N | x_{N-1}) + \ln L(d_N | x_N). \quad (13)$$

The transition probabilities, $p(x_k | x_{k-1})$, can be calculated using a distance rule such as the one discussed above, while incremental data log-likelihoods, $\ln L(d_k | x_k)$ can be calculated by observing that the pixel grey level values g_{ij} are $N(r_{in}, \sigma^2)$ if g_{ij} lies inside the region and $N(r_{out}, \sigma^2)$ when g_{ij} lies outside the region. Furthermore once x_k has been specified, all pixel values g_{ij} in d_k can be associated with pixels either inside or outside of the region. Hence if we define

$$P_G(x) \triangleq \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-x^2/2\sigma^2)$$

we can use

$$\begin{aligned} \ln L(d_k | x_k) &= \sum_{\substack{g_{ij} \in d_k \\ (i,j) \text{ inside} \\ \text{region}}} \ln P_G(g_{ij} - r_{in}) + \sum_{\substack{g_{ij} \in d_k \\ (i,j) \text{ outside} \\ \text{region}}} \ln P_G(g_{ij} - r_{out}) \\ &= C - \sum_{\substack{g_{ij} \in d_k \\ (i,j) \text{ inside} \\ \text{region}}} (g_{ij} - r_{in})^2 / 2\sigma^2 - \sum_{\substack{g_{ij} \in d_k \\ (i,j) \text{ outside} \\ \text{region}}} (g_{ij} - r_{out})^2 / 2\sigma^2 \end{aligned} \quad (14)$$

where C is a constant which is independent of the choice of x_k .

Finally, a dynamic programming algorithm for estimating a state sequence S_N and hence a boundary edge sequence $\{t_i\}_1^M$ which maximizes $\ln L(D_N, S_N)$ can be derived by observing that

$$\max_{x_n \in S_{N-1}} \ln L(D_N, S_N) = \max[\max \ln L(D_{N-1}, S_{N-1}) + \ln p(x_N | x_{N-1}) \\ + \ln L(d_N | x_N)] \quad (15)$$

This completes the specification of the algorithm.

References

- [1] A. S. Willsky, "Relationships between Digital Signal Processing and Control and Estimation Theory," Proc. IEEE, 66, pp. 996-1017 (September 1978).
- [2] T. Kailath, "A View of Three Decades of Linear Filtering Theory," IEEE Trans. Inform. Theory, vol. IT-20, no. 2, pp. 146-181 (March 1974).
- [3] A. J. Viterbi, "Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm," IEEE Trans. Inform. Theory, IT-13, pp. 260-269 (April 1967).
- [4] C. R. Cahn, "Phase Tracking and Demodulation with Delay," IEEE Trans. Inform. Theory, IT-20, pp. 50-58 (January 1974).
- [5] G. D. Forney, Jr., "The Viterbi Algorithm," Proc. IEEE, 61, pp. 268-278 (March 1973).
- [6] J. P. Dugré, L. L. Scharf, and A. A. Beex, "Generating Covariance Sequences and the Calculation of Quantization and Rounding Error Variances in Digital Filters," IEEE Trans. ASSP, to appear.
- [7] L. L. Scharf and L. W. Nolte, "Likelihood Ratios for Sequential Hypothesis Testing on Markov Sequences," IEEE Trans. on Inform. Theory, vol. IT-23, no. 1, (January 1977).
- [8] R. A. Bellman, "Introduction to Matrix Analysis," McGraw-Hill, New York, pp. 155-156 (1970).
- [9] L. L. Scharf, D. D. Cox, and C. J. Masreliez, "Modulo- 2π Phase Sequence Estimation," IEEE Trans. Inform. Theory, in print.
- [10] M. H. Hueckel, "A Local Visual Operator which Recognizes Edges and Lines," Journal of the ACM, 20, (October 1973).
- [11] H. Elliott, D. Cooper, and P. Symosek, "Implementation, Interpretation and Analysis of a Suboptimal Boundary Finding Algorithm," Proceedings of the 1979 IEEE Computer Society Pattern Recognition and Image Processing Conference, Chicago, (August 1979).

Figure Captions

- Figure 1.** States and Characters
- Figure 2.** Sunflowers, States, and Characters for Frequency Tracking
- Figure 3.** Goertzel Filter for DFT Component X_m
- Figure 4.** Four Point Decimation-in-Frequency FFT
- Figure 5.** Prediction, Detection, and Control of a Noisy AR(N) Sequence
- Figure 6.** Visualizing the Random Walk Frequency Trajectories
- Figure 7.** Date Processing and Frequency Trellis Illustrating Evolution of Surviving Frequency Tracks
- Figure 8.** A Boundary Segment in Small Picture Segment
- Figure 9.** Example of Boundary Segment Generation
- Figure 10a.** Example of a Boundary Consistent with Model
- Figure 10b.** Example of a Boundary Inconsistent with Model
- Figure 11.** CAT Scan of Lung Section
- Figure 12.** Possible State Locations x_k^j for $k=1,2,\dots,5$
- Figure 13.** State Transition Diagram for $k=1,2,\dots,5$ Illustrating a Set of Characters C_k , $k=1,2,\dots,5$ for a Specific Process Realization x_k , $k=1,2,\dots,5$

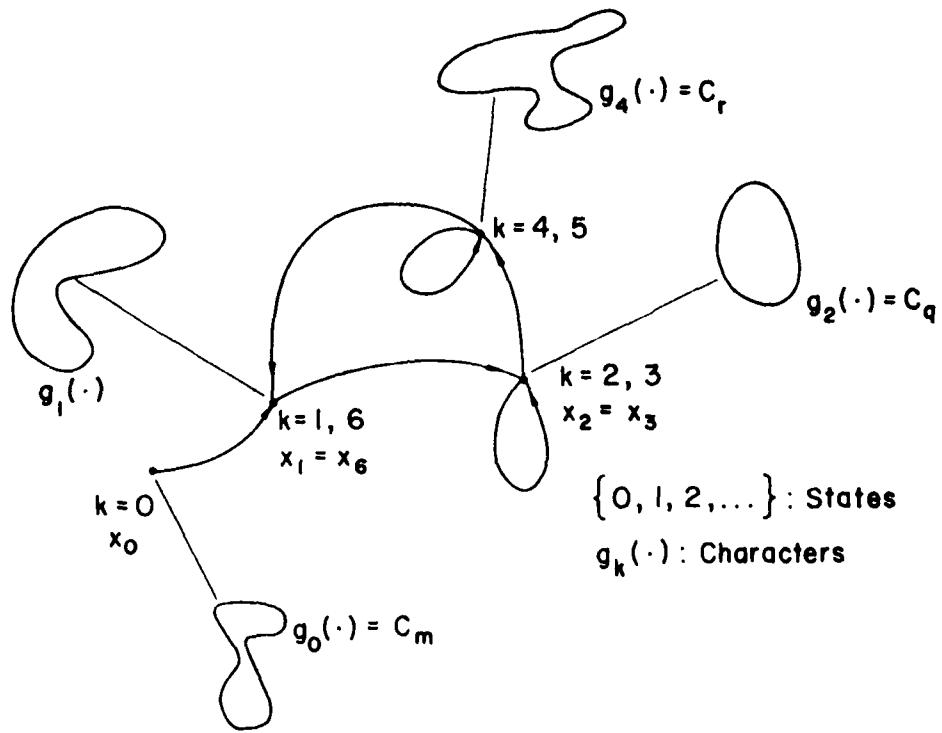


Figure 1. States and Characters

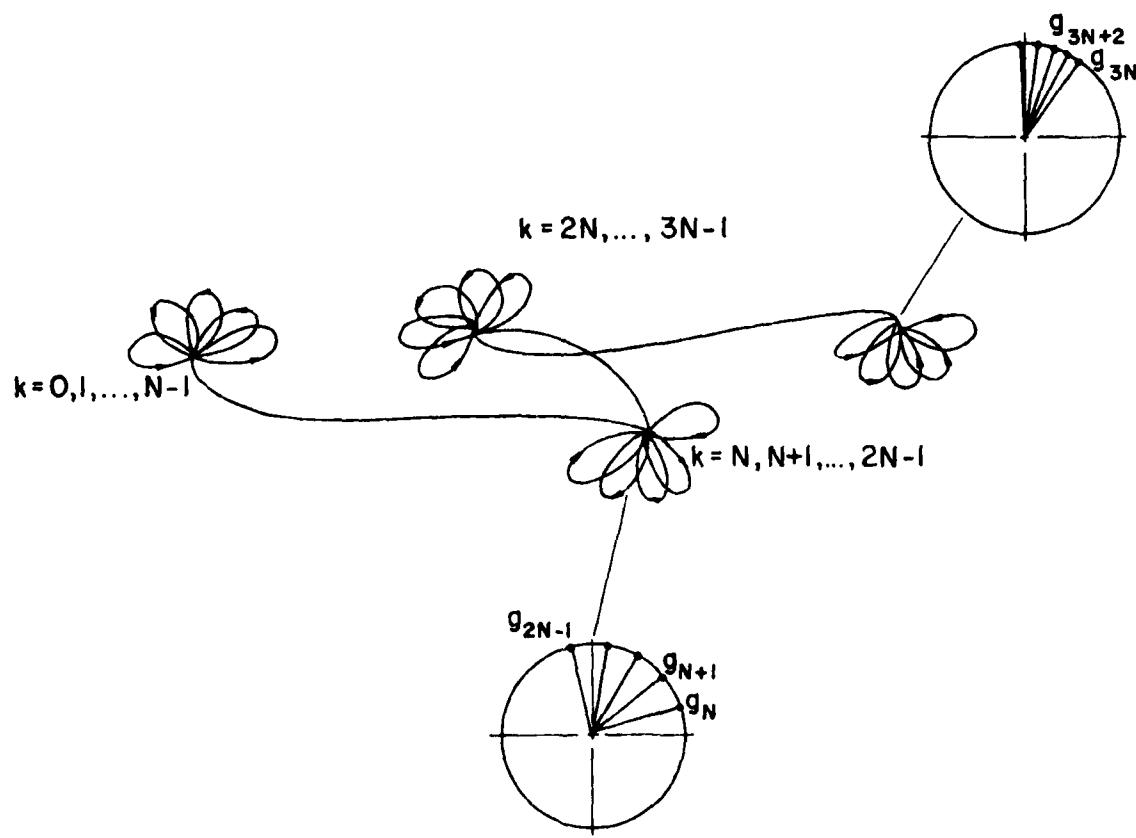


Figure 2. Sunflowers, States, and Characters for Frequency Tracking

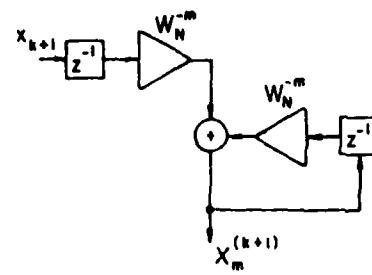


Figure 3. Goertzel Filter for DFT Component X_m

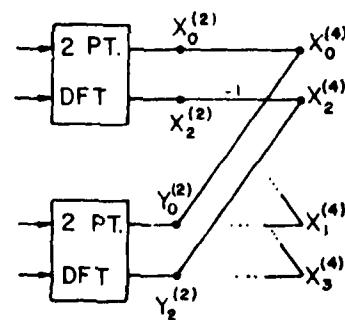


Figure 4. Four Point Decimation-in-Frequency FFT

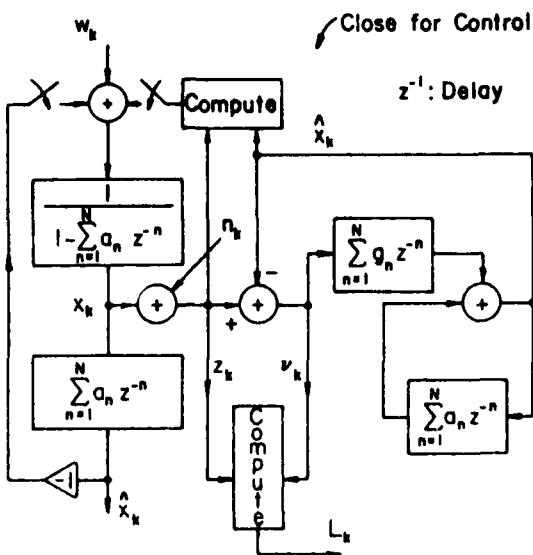


Figure 5. Prediction, Detection, and Control of a Noisy AR(N) Sequence

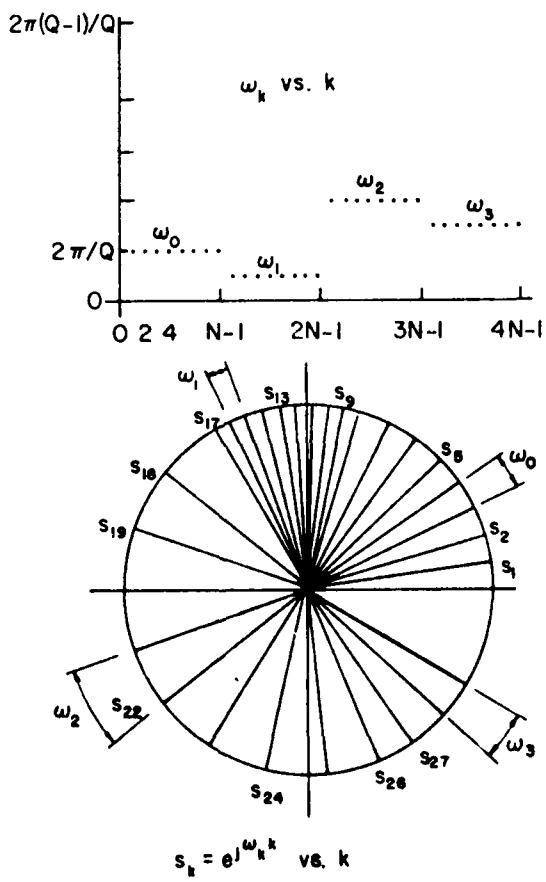


Figure 6. Visualizing the Random Walk Frequency Trajectories

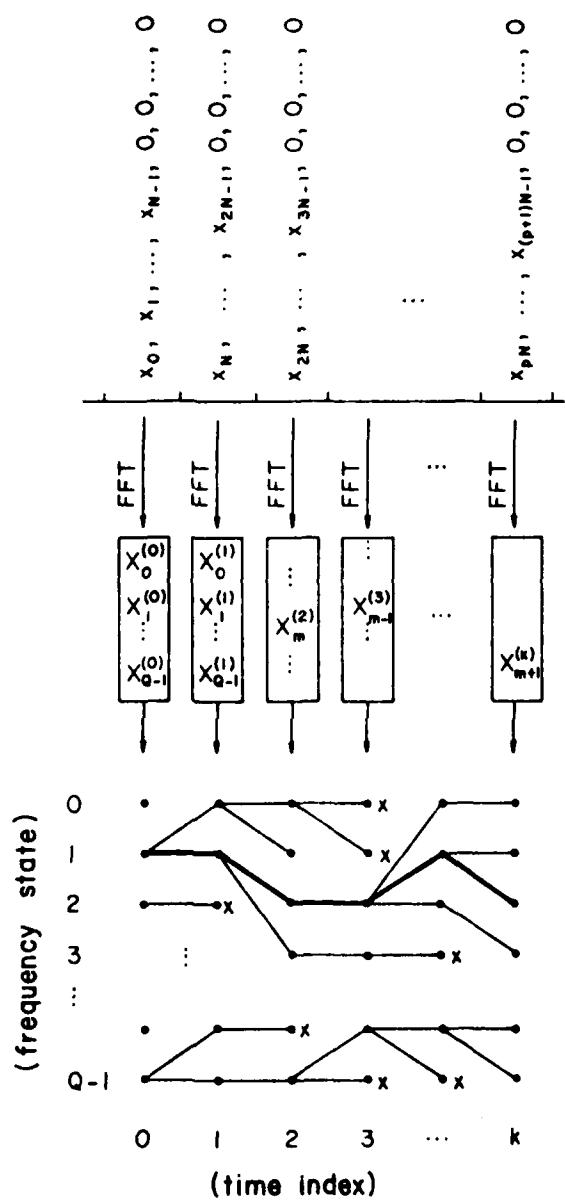


Figure 7. Data Processing and Frequency Trellis Illustrating Evolution of Surviving Frequency Tracks

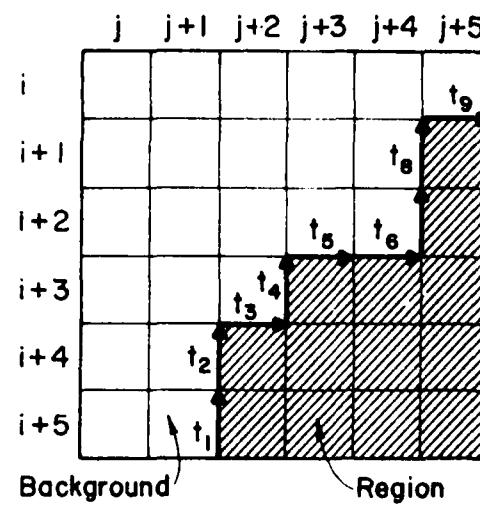


Figure 8. A Boundary Segment in Small Picture Segment

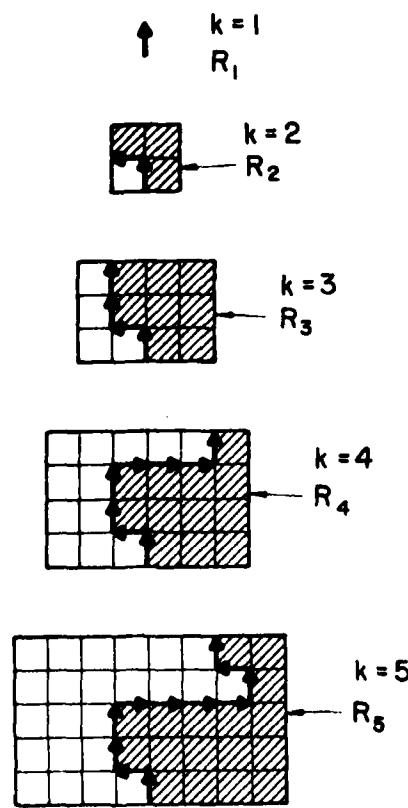


Figure 9. Example of Boundary Segment Generation

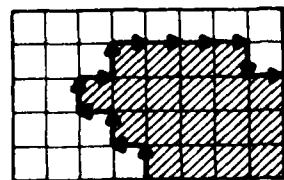


Figure 10(a). Example of a Boundary Consistent with Model

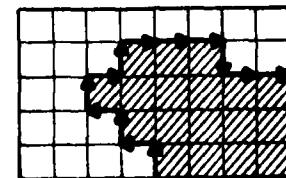


Figure 10(b). Example of a Boundary Inconsistent with Model



Figure 11. CAT Scan of Lung Section

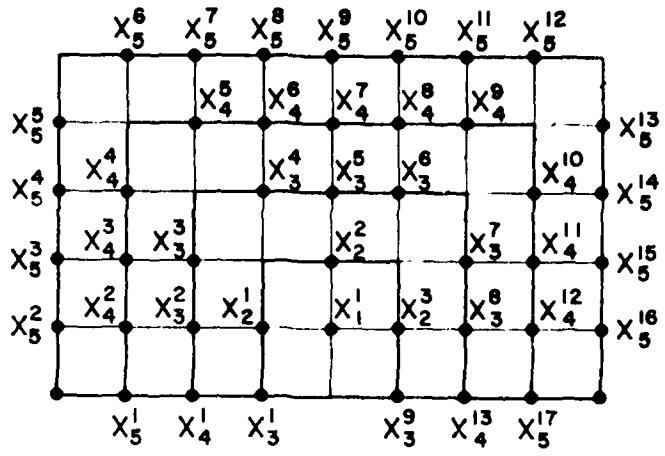


Figure 12. Possible State Locations x_k^j for $k=1,2,\dots,5$

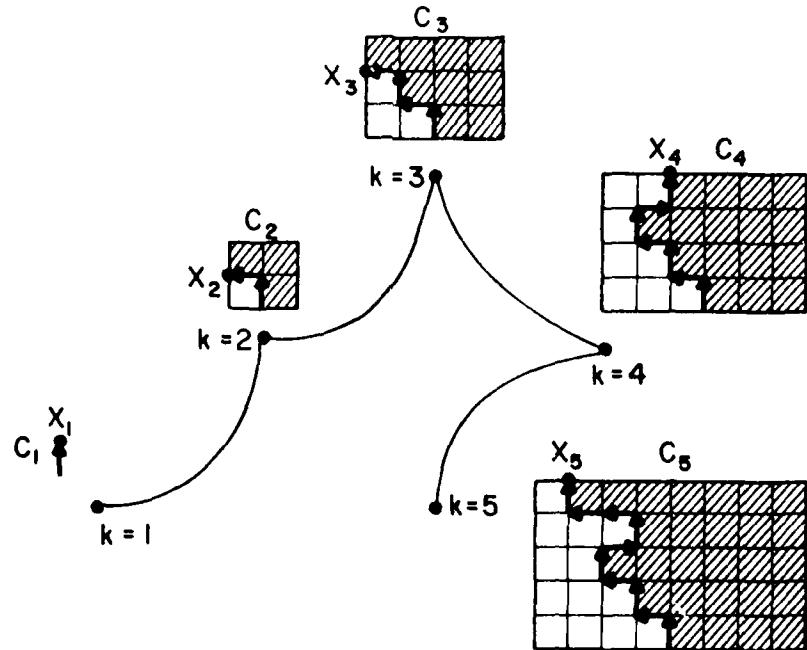


Figure 13. State Transition Diagram for $k=1,2,\dots,5$ Illustrating a Set of Characters C_k , $k=1,2,\dots,5$ for a Specific Process Realization x_k , $k=1,2,\dots,5$